Introduction to Distributed Systems

Brian Nielsen

bnielsen@cs.aau.dk

About me

- Brian Nielsen

- Lecturer in Computer Science
- Distributed and Embedded Systems

- Research
  - Distributed programming (Linda w. multiple tuple spaces)
  - Distributed Multi-media
  - Real-Time Operating Systems
  - Automated testing of embedded, real-time, distributed systems

About You

- Dat-3
- F7S
- SSE-1 / MI1 / DE-1
- Guests (SSE-1)
- EVU
- Project Course vs Study Course
- $\Rightarrow$ Different background and expectations

Teaching Assistant

- Claus Rørbæk Thrane:crt@cs.aau.dk

Study Regulations

*“**Purpose**: That the student obtains knowledge about concepts in distributed systems, knowledge about their construction, and an understanding of advantages and disadvantages of their use.”*

Study Regulations

Contents:

- Structure of distributed systems.
- Distributed algorithms.
- Distributed and parallel programming.
- Fault tolerance.
- Examples of one or more distributed systems.

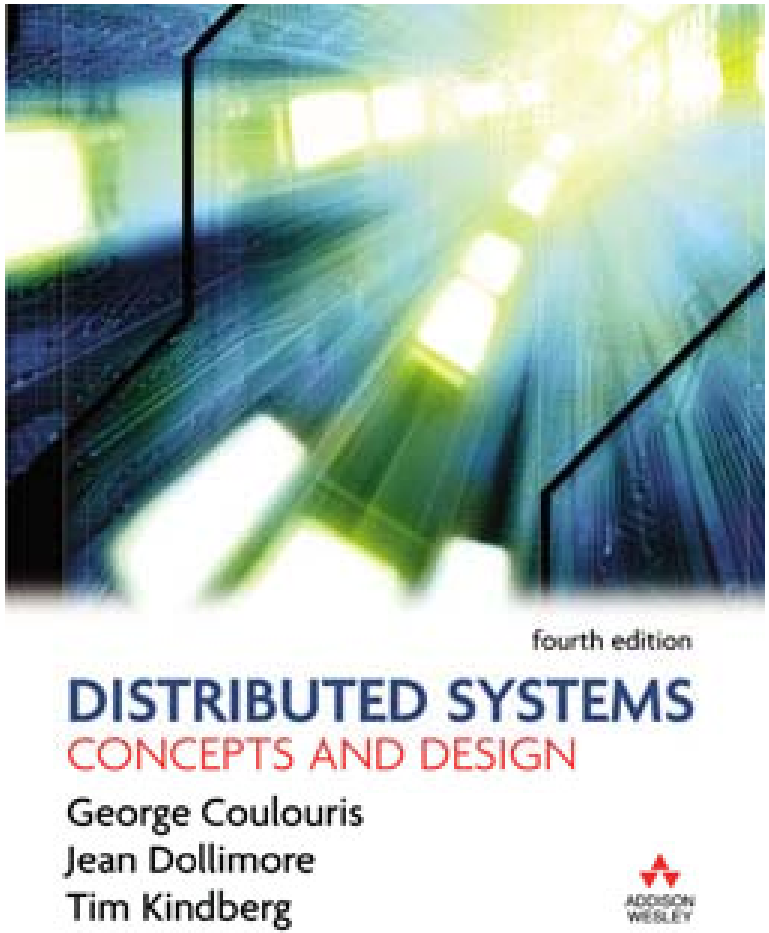
Goals

- Solid understanding of the characteristics and properties of Distributed System
- Solutions
  - Programming languages/models
  - Middleware
  - Distributed algorithms for typical problems: mutex, multicast, consensus, replication
- Advantages and disadvantages of a distributed solution
  - When and which to choose
  - Evaluate a system design
- Skill in implementation of distributed systems/algorithms

Course Form

- 3 ECTS (3*30 study hours), 15 lectures
- 1 lecture = 6 study hrs
  - 2*45 min of lectures
  - 1.5 hrs of exercises with TAs in groups
  - 1.5 hrs of reading homework
  - 0.5 hrs of exam-preparation
- 1 “big” study assignment subject to examination
- PE: exam part of project-exam
- SE: Oral 20 mins, Binary scale!

Text Book



- Coulouris, Dollimore and Kindberg
- **Distributed Systems: Concepts and Design**
- Edition 4
- www.cdk4.net

Preliminary Course Plan

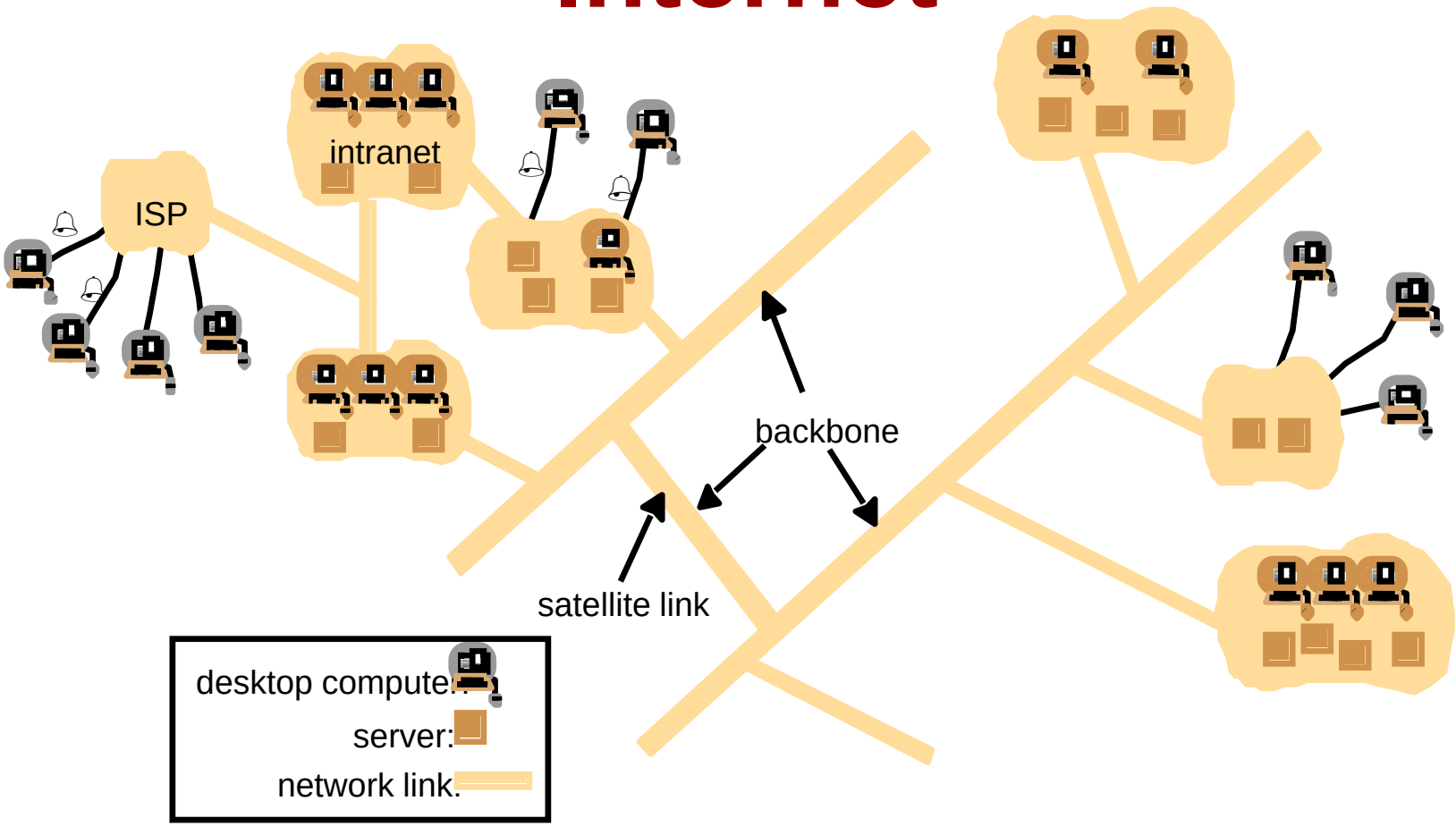
Lecture	Topic
1	**Introduction to Distributed Systems**
2	**Programming Models I - RPC, RMI**
3	**Programming Models II – message passing, events, shared memory**
4	**Distributed File Systems**
5	**Peer2peer Systems**
6	**Clock Synchronization**
7	**Distributed Mutual Exclusion & Election**
8	**Multicast communication**
9	**Consensus and study-exercises**
10	**Replication**
13	**Study Exercise**
11	**Web Services**
12	**Introduction to Grid Computing (Guest Lecture by Josva Kleist)**
14	**Study Exercise**
15a	**Conclusions and Exam Information**
15b	**Exam Questioning Hour / Spørgetime**

Pre-requisites

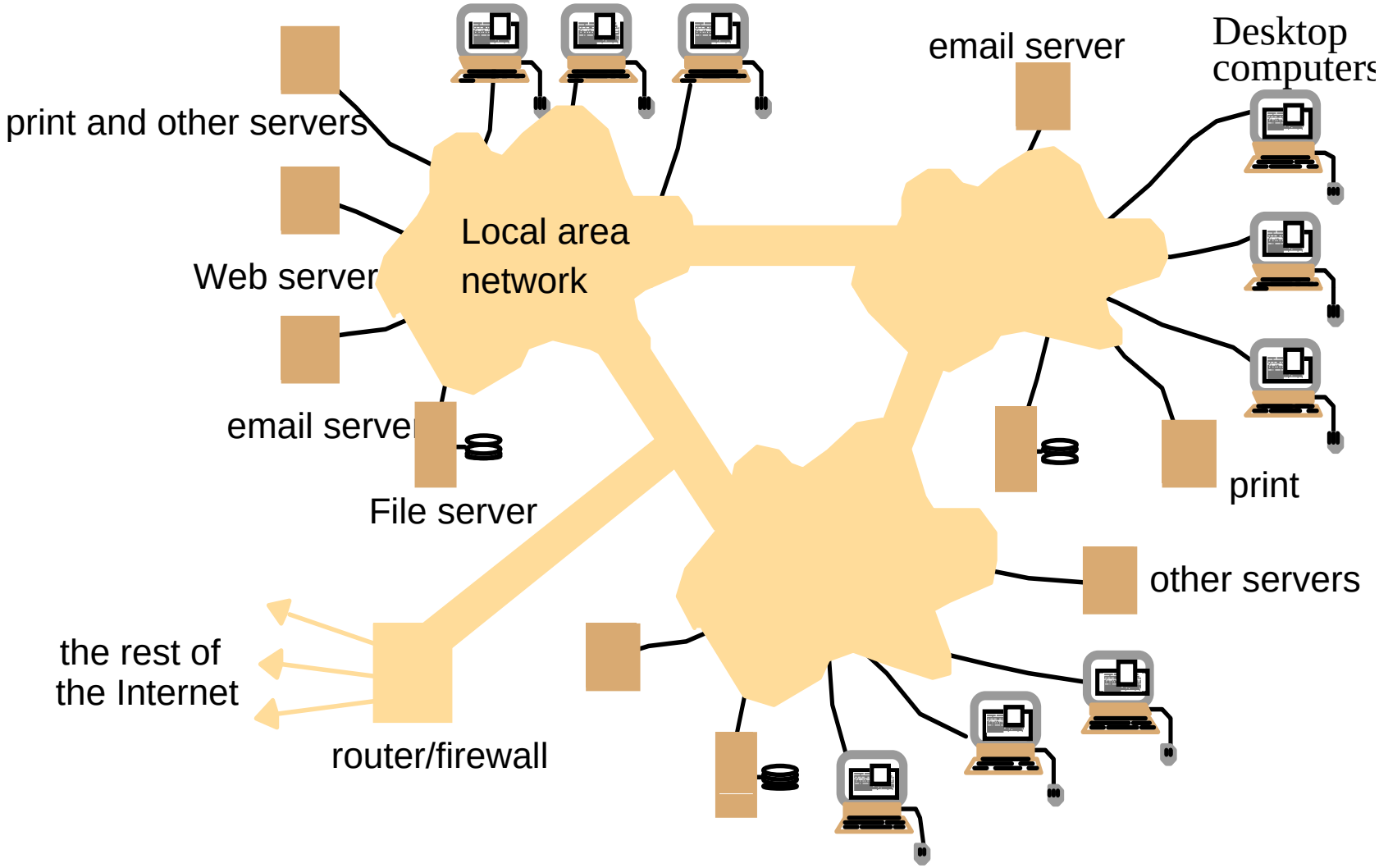
- Programming
  - Practical programming in e.g. Java, C, C++
  - Basic data-structures and algorithms
  - Preferably also concurrent programming
- Networks
  - IP-stack, IP-addressing, IP-routing, message enveloping, TCP/UDP, Sliding-window, socket-programming, basic encryption technology
  - Else read chapter 3
- Operating Systems
  - Processes, threads, concurrency, non-determinism, kernel and user level, synchronization (semaphores, monitors), address spaces, virtual memory, file-systems
  - Else read chapter 6

Examples of Distributed Systems

A typical portion of the Internet



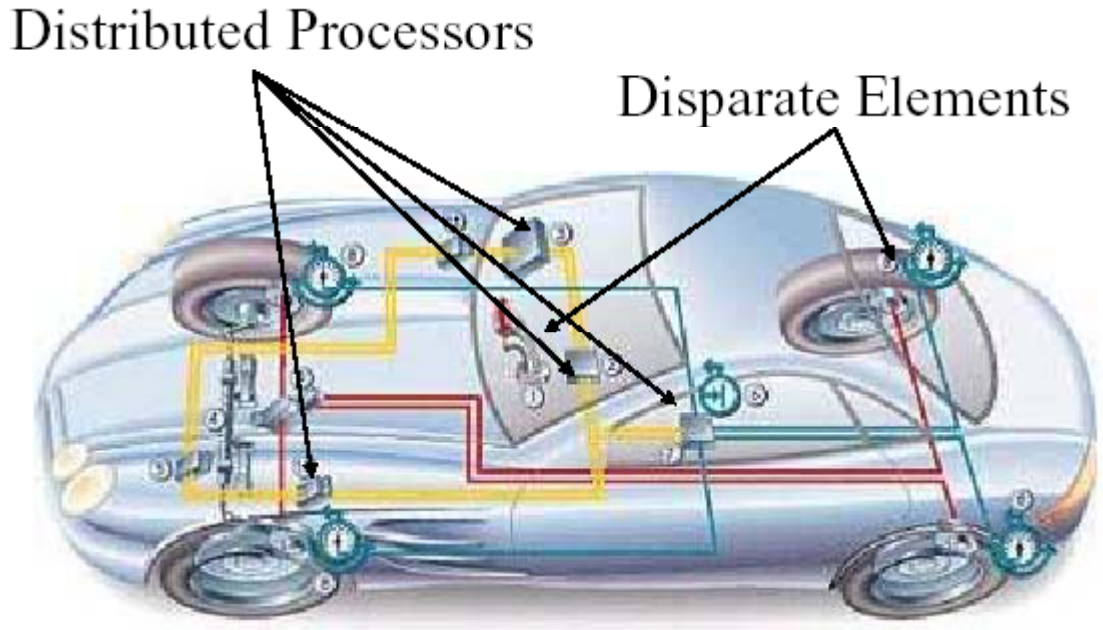
A typical intranet



Intranets

- A portion of the Internet that
  - is separately administered
  - usually proprietary
  - provides internal and external services
  - can be configured to enforce local security policies
    - may use a firewall to prevent unauthorized messages leaving or entering
  - may be connected to the internet via a router
- Services:
  - File, print services, backup, program-sharing, user-, system-administration, internet access

Automotive Control



- 80+ ECU's interconnected in controller area networks
- Vehicle dynamics (engine, brake, gear control,…)
- Instrumentation control (lights, indicators, windows,…)
- System Integration
- Information and entertainment systems
- Drive-by-wire

Distributed Computing

- Speed up huge computations by using multiple computers
- NOWs (network of workstations) / cluster computing

- Dedicated computers
- seti@home: project to scan data retrieved by a radio telescope to search for radio signals from another world;
- Grid Computing: www.grid.org (Millions CPUs world-wide – source 2004)

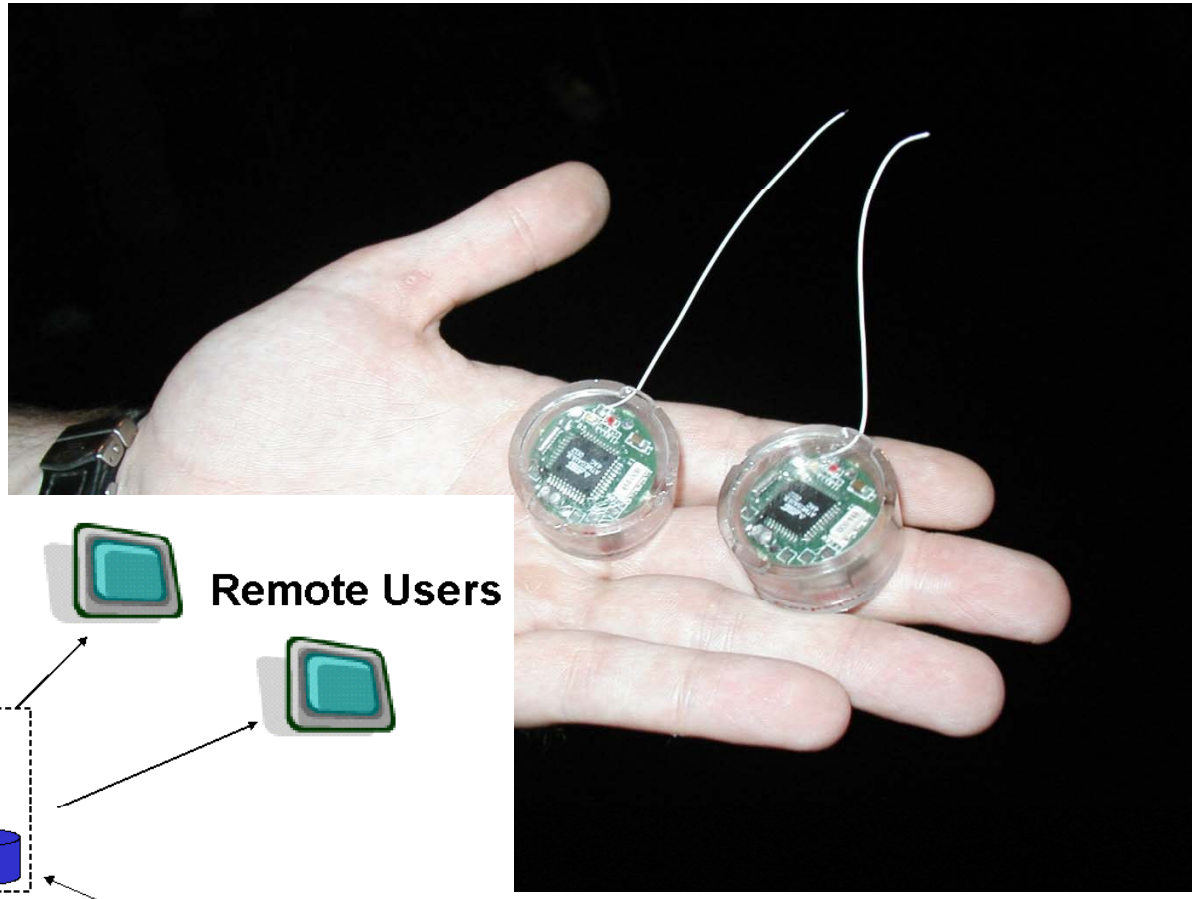
Cluster & Grid Computing

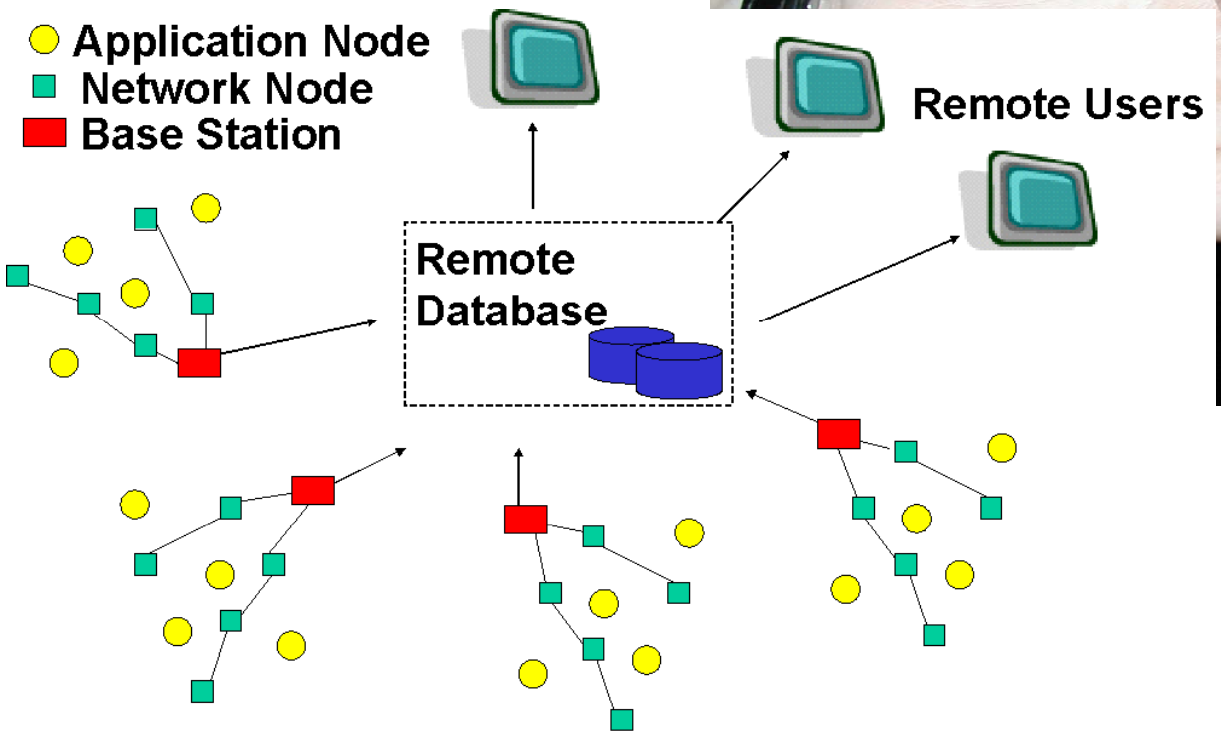
Home made PC cluster

Fyrkat@AAU

Fyrkat is a 672 core cluster with specs:
84 IBM blades - 2 Xeon quad core 2 cpus 2.33 Ghz
16 GB memory per blade
Infiniband interconnect

Sensor-networks





Games


Piken Square (American District 2)
Level: 10
Party Formation
Members (1/4)
N/E10 Malekith Shanar
Mlana Bloodmoon: Where on the map is the courthouse?
Hades Ariaenor: looking for party for garfazz bloodfang!
Sir Zan: RUNNING TO GRENDITCH COURTHOUSE 300 GOLD (FAST TRAVEL GAURENTEED ARRIVAL)
Darth Fenix: SELLING LONGBOW OF FORTITUDE BY STORAGE
280
36
Menu

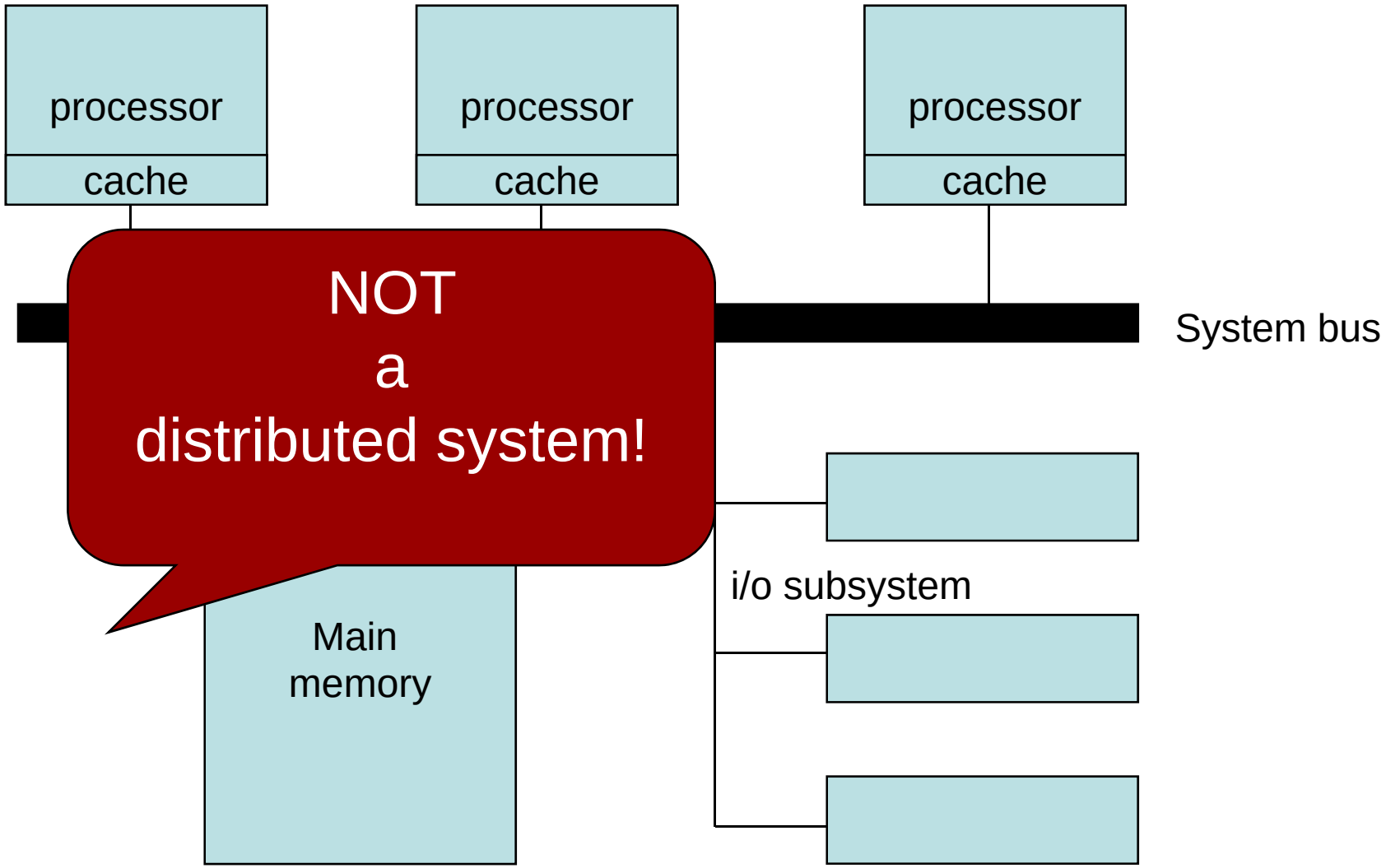
CLIENT
CLIENT
SERVER
CLIENT

CLIENT
CLIENT
CLIENT
CLIENT

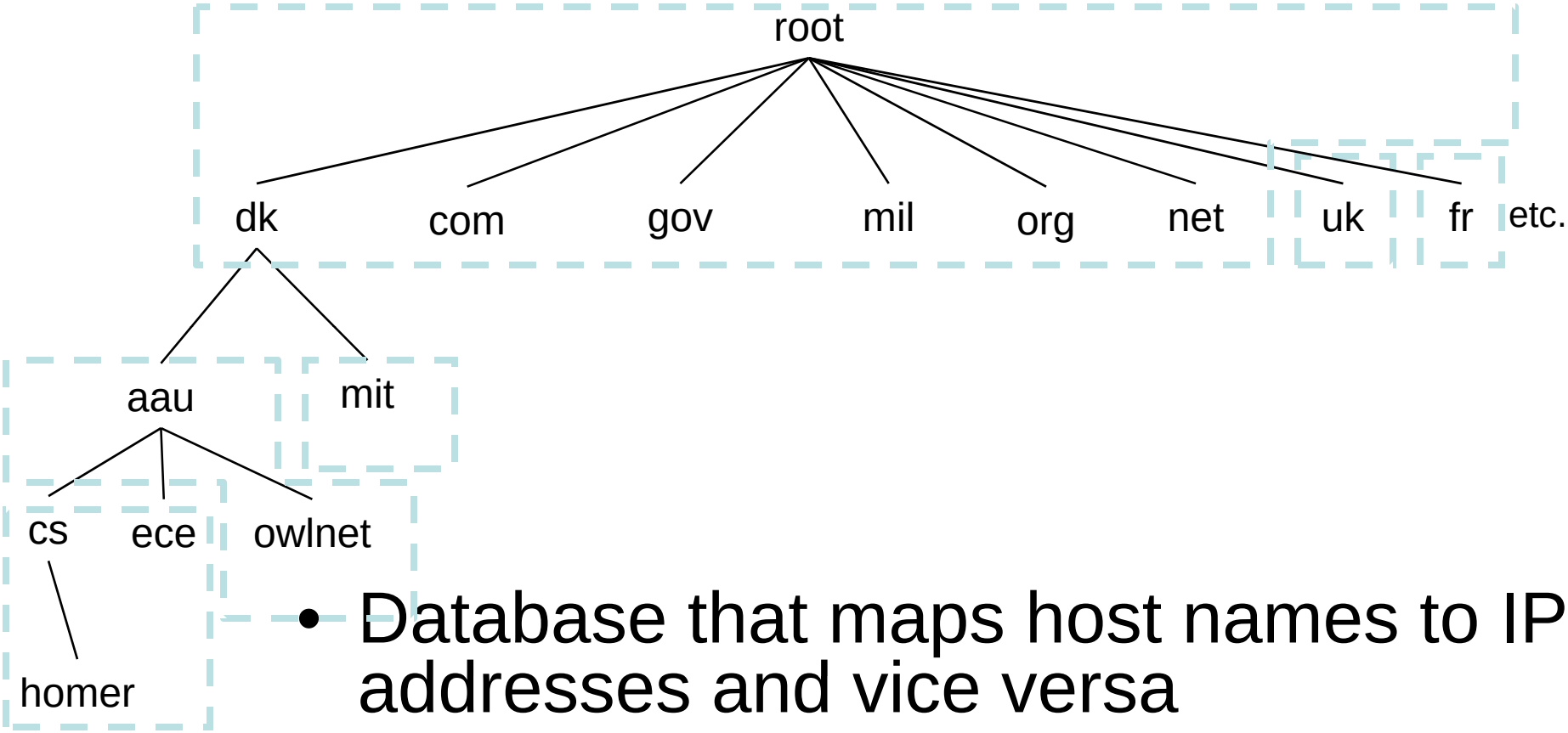
Mission-critical applications

- Embedded systems, automotive, avionics
- Control Systems
- Banking, stock markets, stock brokerages
- Health care, hospital automation
- Control of power plants, electric grid
- Telecommunications infrastructure
- Electronic commerce and electronic cash on the Web (very important emerging area)
- Corporate “information” base: a company’s memory of decisions, technologies, strategy
- Military command, control, intelligence systems
- …

Shared Memory Multi-Processor



Domain Name Service



- Database that maps host names to IP-addresses and vice versa
- homer.cs.aau.dk =130.225.194.13
- Client--server interaction on UDP Port 53

DNS: History

- Initially all host-addess mappings were in a file called hosts.txt (in /etc/hosts)
  - Changes were submitted to SRI (Stanford Research Institute) by email
  - New versions of hosts.txt ftp'd periodically from SRI
  - An administrator could pick names at their discretion
  - Any name is allowed: eugenesdesktopatrice (flat namespace)
- As the internet grew this system broke down because:
  - SRI couldn't handled the load
  - Hard to enforce uniqueness of names
  - Many hosts had inaccurate copies of hosts.txt
- Domain Name System (DNS) was born in '83

DNS zone data records

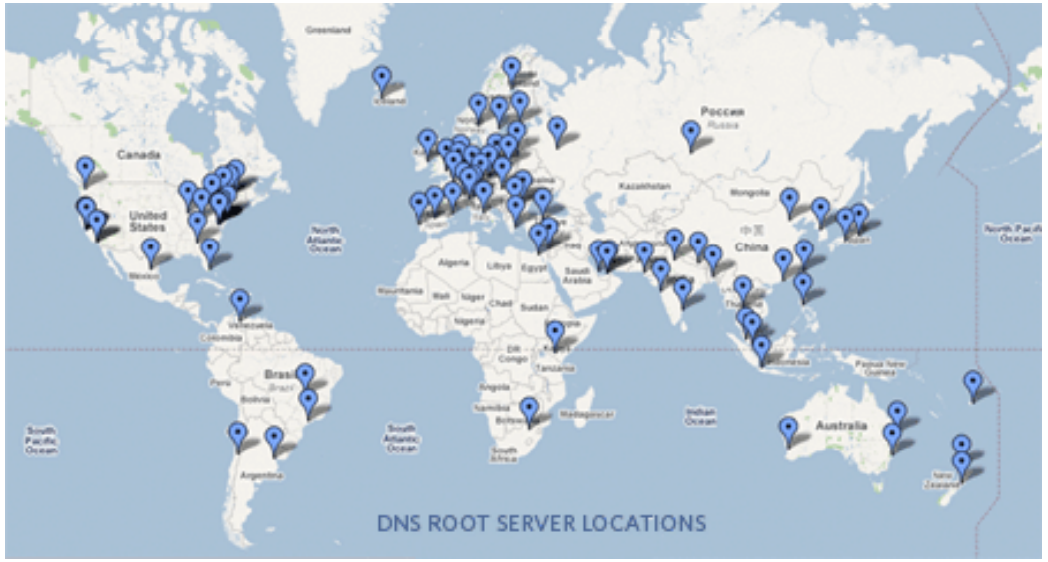
Example Domain dcs.qmul.ac.uk

domain name	*time to live*	*class*	*type*	*value*
www	1D	IN	CNAME	apricot
apricot	1D	IN	A	138.37.88.248
dcs	1D	IN	NS	dns0.dcs
dns0.dcs	1D	IN	A	138.37.88.249
dcs	1D	IN	NS	dns1.dcs
dns1.dcs	1D	IN	A	138.37.94.248
dcs	1D	IN	NS	cancer.ucs.ed.ac.uk

CNAME = Canonical name for an alias
A = IP Address
NS = Authorative name server

DNS: Root Name Servers

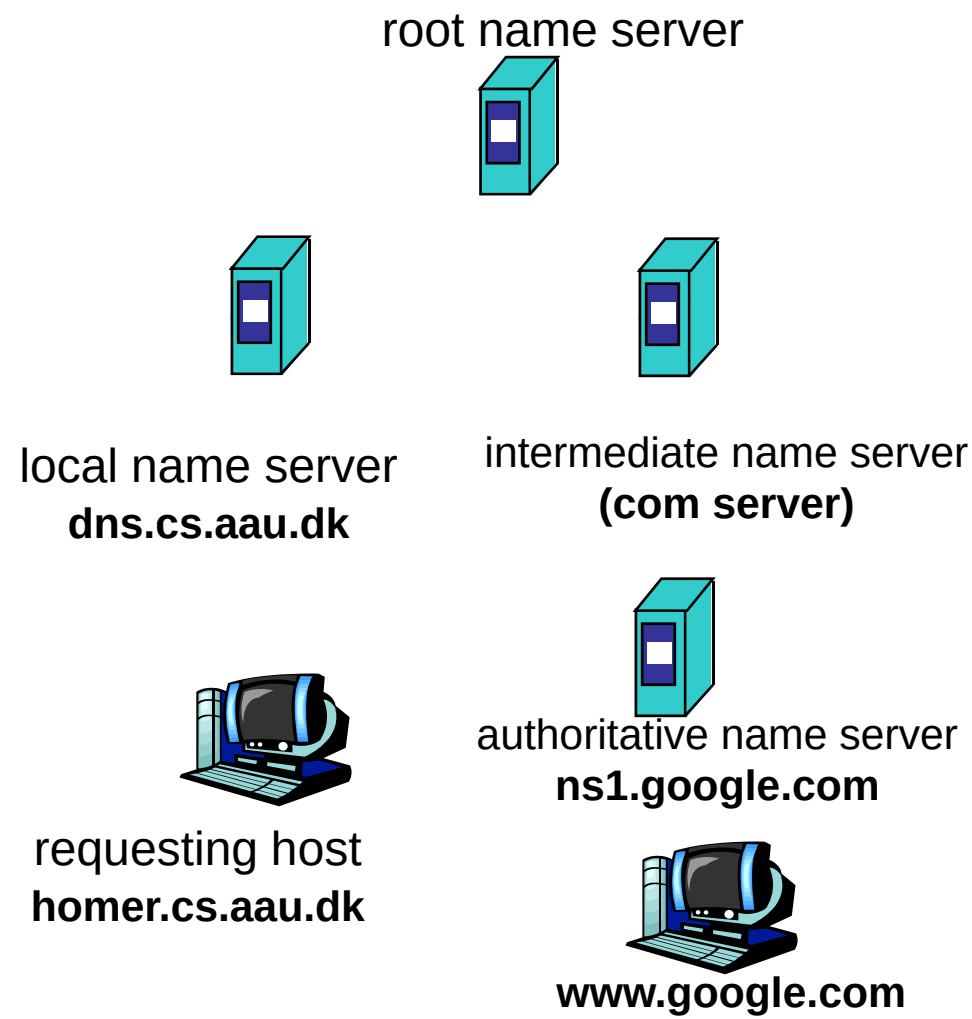
- Contacted by local name server that can not resolve name
- Root name server:
  - Contacts authoritative name server if name mapping not known
  - Gets mapping
  - Returns mapping to local name server
- **Several** root name servers worldwide



DNS Servers

Root name server:

- May not know authoritative name server
- May know intermediate name server: who to contact to find authoritative name server?



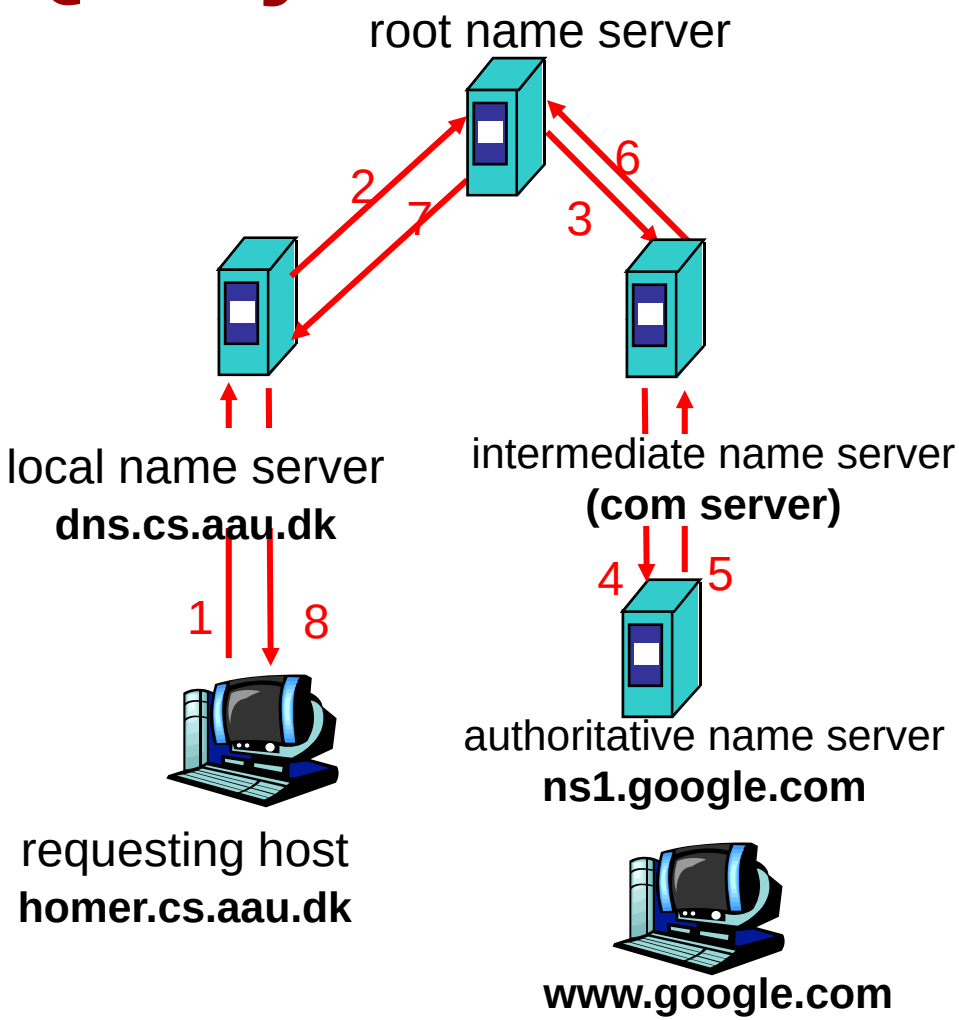
Example of Recursive DNS Query

Root name server:

- May not know authoritative name server
- May know intermediate name server: who to contact to find authoritative name server?

Recursive query:

- Puts burden of name resolution on contacted name server
- Heavy load?

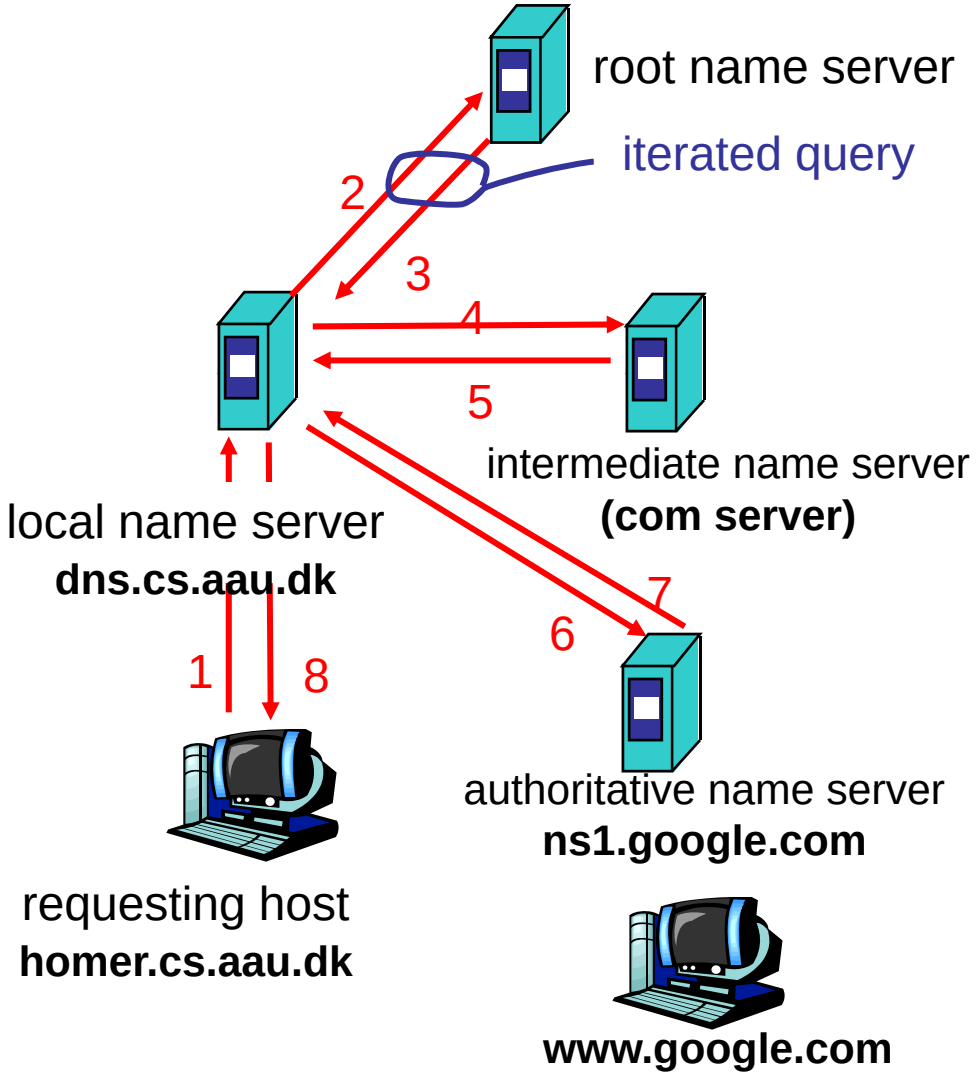


Example of Iterated DNS Query

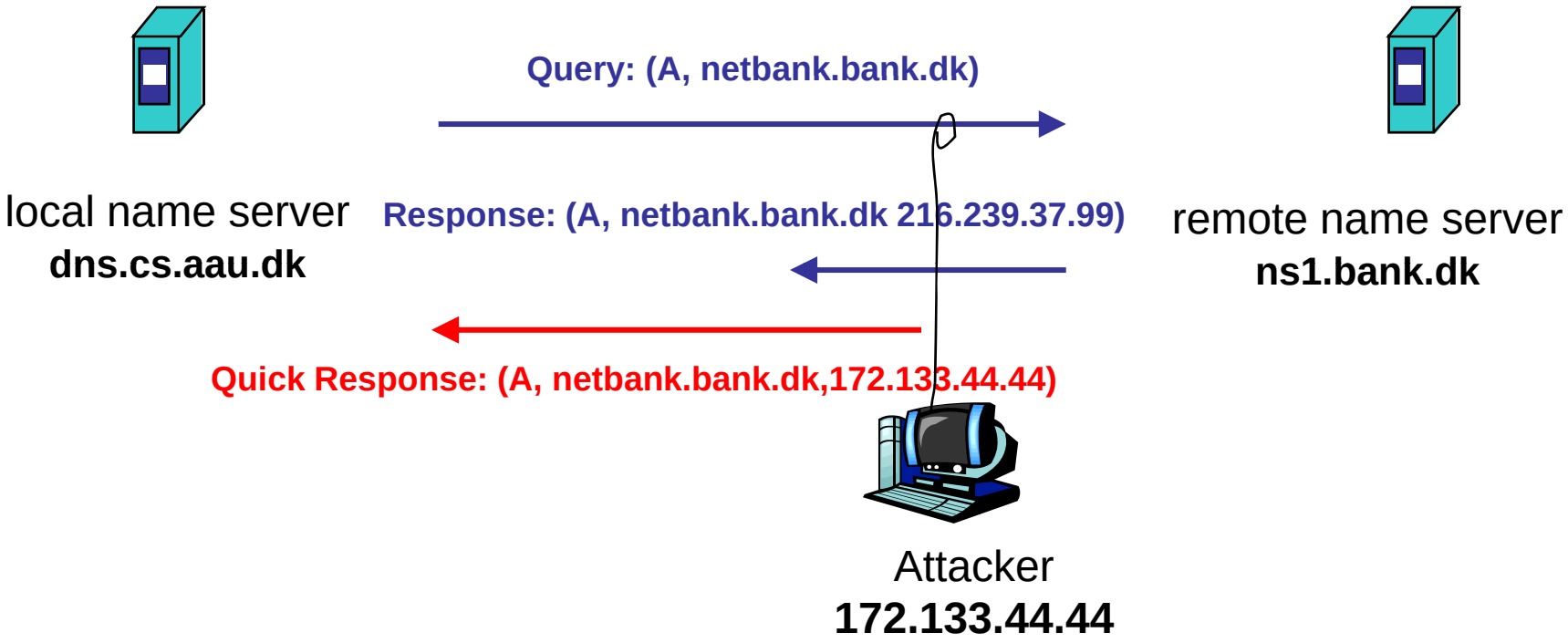
Iterated query
  Contacted server replies with name of server to contact

- "I don't know this name, but ask this server" (delegation):

This is how today's DNS system behaves



DNS-spoofing



- Until the TTL expires, 172.133.44.44 serves netbank.dk

Lessons Learned

- ***Flexibility***
  - *Manual or static configuration or update is too inflexible*
  - *Must work across organizations and administrative domains with different security and resource usage policies*
  - *Heterogeneous Computer and OS*
- ***Performance***
  - *Avoid Bottlenecks - typically occurs at centralized components*
  - *Localize access*
    - *to increase response time and minimize (long-distance communication).*
    - *caching*
  - *Distribute load evenly*
  - *Use parallelism for increased processing power.*
  - *Scaleability*
- ***Dependability***
  - *Replication for increased availability, fault tolerant*
  - *Security is always a problem!*
  - *Correct functionality*

Definition

Definition

- *A **distributed system** is the one in which hardware and software components at **networked computers** communicate and coordinate their activity only by **passing messages.***
- Examples: Internet, intranet and mobile computing systems.

Consequences

- Concurrent execution of processes
  - Users work independently & share resources
  - non-determinism, race-conditions, synchronization, deadlock, liveness, …
- No global clock
  - Coordination is done by message exchange
  - There are limits to the accuracy with which computers in a network can synchronize their clocks
- No global state
  - Generally, there is no single process in the distributed system that would have a knowledge of the current global state of the system
- **Units may fail independently.**
  - Network faults can result in the isolation of computers that continue executing
  - A system failure or crash might not be immediately known to other systems

Why a Distributed System?

- Functional distribution
  - computers have different functional (eg. File server, print,) capabilities yet may need to **share resources**
    - Client / server
    - Data gathering / data processing
- Inherent distribution in application domain
    - physically or across administrative domains
    - cash register and inventory systems for supermarket chains
    - computer supported collaborative work
- Economics
  - collections of microprocessors offer a better price/ performance ratio than large mainframes

Why a Distributed System?

- Better performance
  - Load balancing
  - Replication of processing power
- Increased Reliability
  - Exploit independent failures property and
  - Redundancy

Models

Architectural

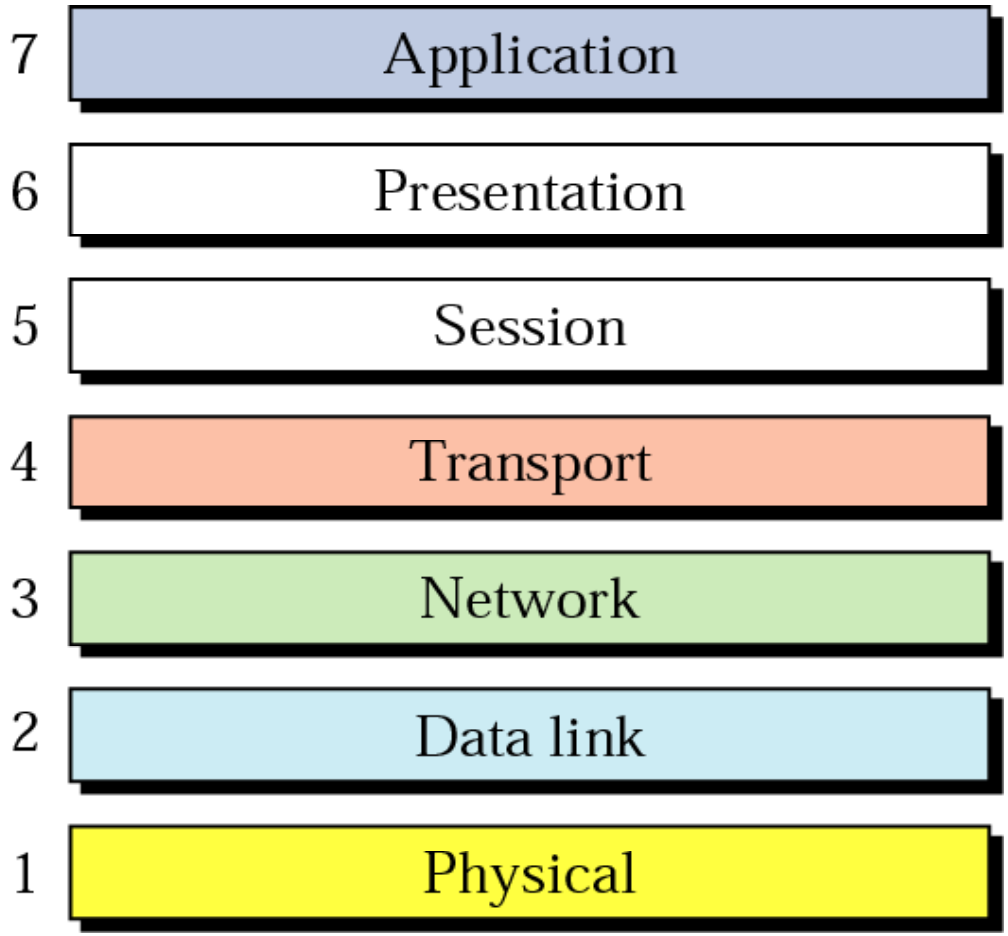
Fundamental (semantic assumptions)

Architectural models

- Software layers
- System architecture

OSI-model

- **Open Systems Interconnection model (ISO standard)**

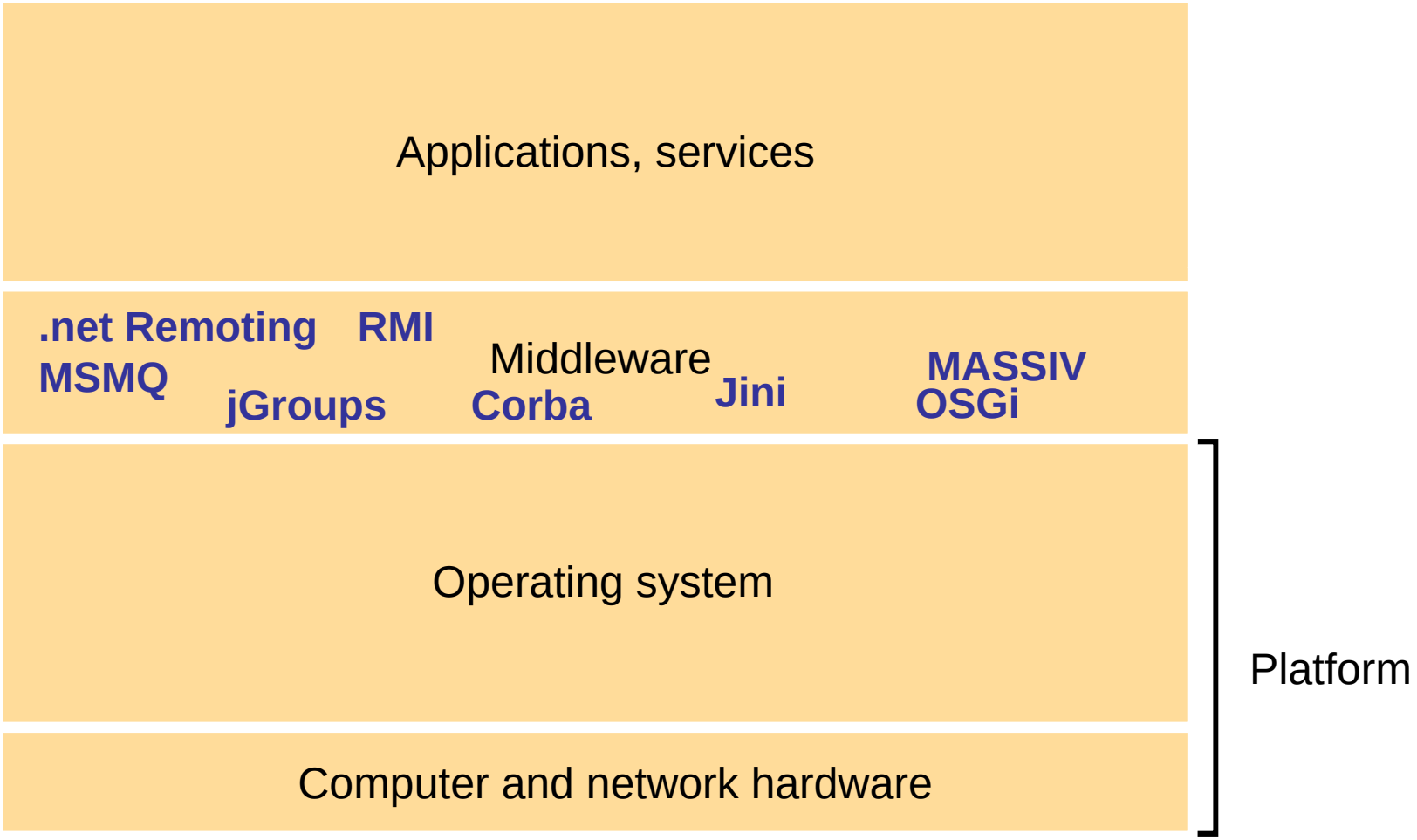


- **Session Layer:** Dialog controller
  - Establish
  - Maintain
  - Synchronize
  - Terminate

Presentation layer: handles syntax and semantics

- Data translation
- Encryption/decryption
- Compression/expansion

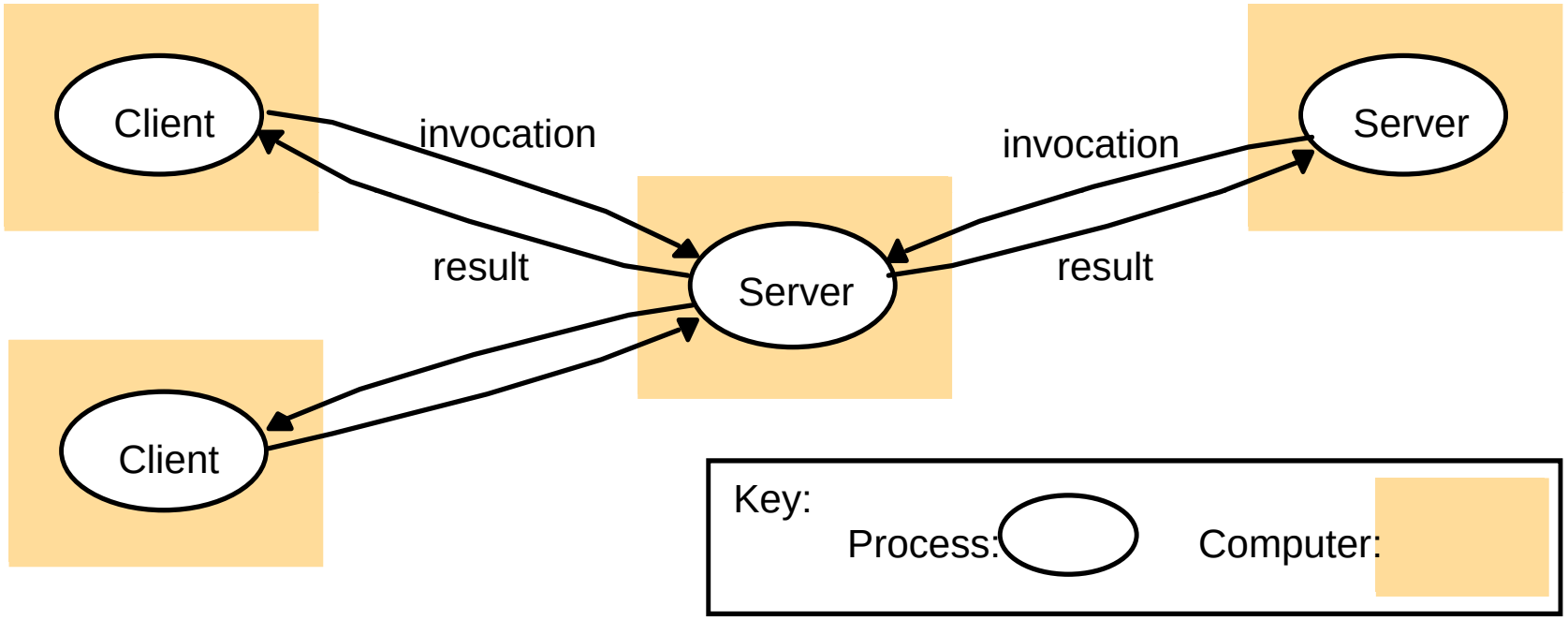
Service layers



Middleware

- Software layer (library of functions) that simplifies programming
  - Masks heterogeneity
  - Provides a convenient programming model
    - Objects/ processes
    - Communication primitives
    - Synchronization
    - Group and multicasting
    - Naming and Localization services
    - Event notification
  - Corba, JavaRMI, .NET Remoting, MPI, ISIS,…

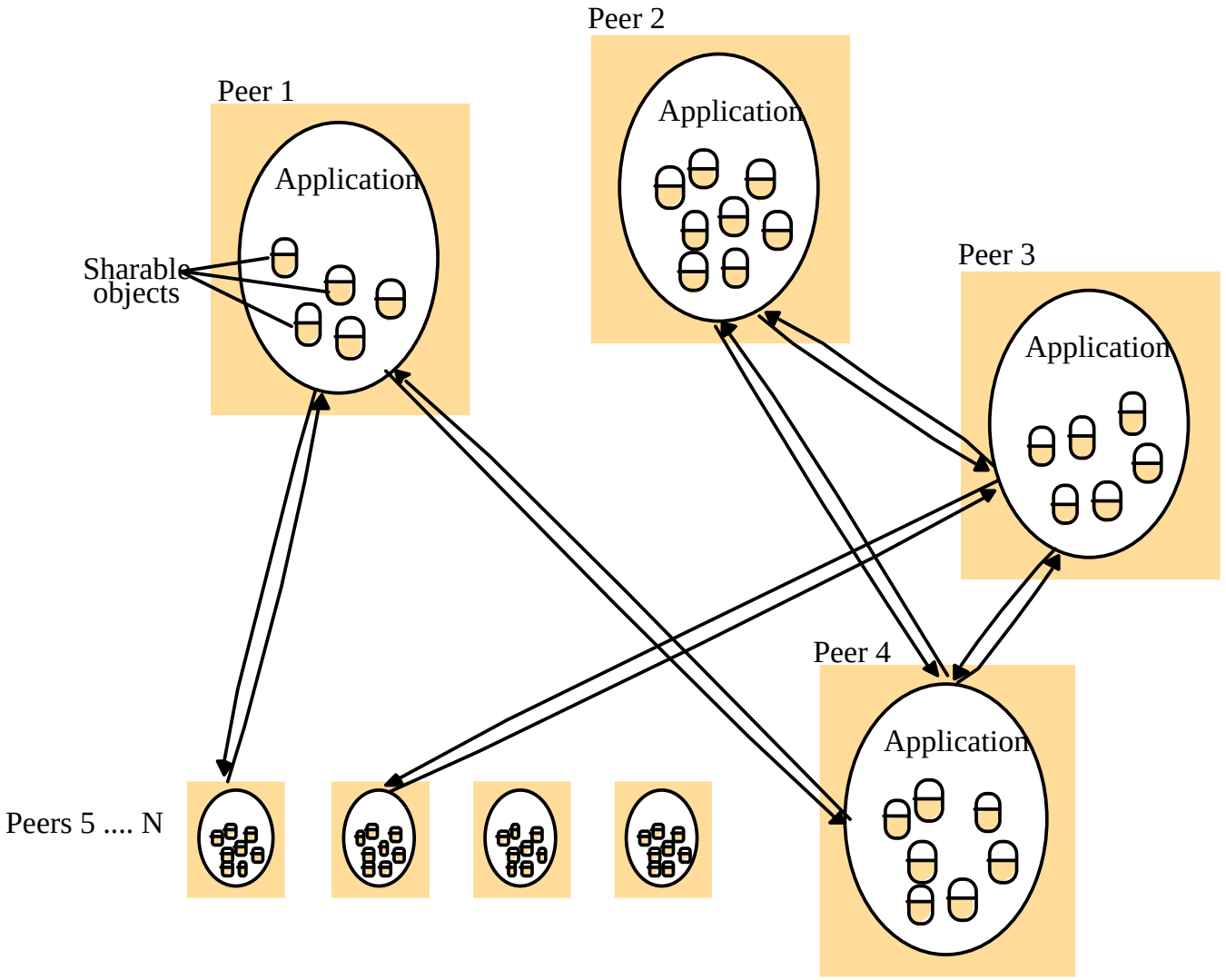
Clients / Server model



Variations:

- thin / thick / smart (dynamic) clients
- Server-farms
- multi-tier systems

A distributed application based on peer processes



Client Server vs Peer based

Client-Server

- Most widely used model
- Functional specialization
- Asymmetrical
- Tends to be Centralized
- Tends to scale poorly

Peer

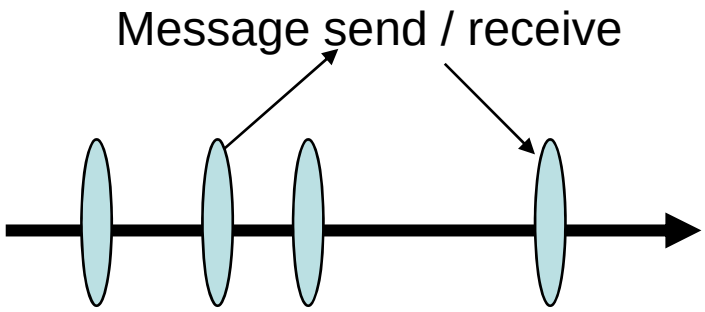
- Symmetrical, computers runs same algorithms / same responsibilities
- Truly Distributed
- Share / exploit resources at a large number of participants

Fundamental Models

Design and solutions depend on fundamental assumptions on

- Process Interaction
- Failures
- Security threats

Interaction Model



- Process:
  - executing program with private state
  - sending and receiving messages
- Distributed Algorithms:
  - *A definition of the steps to be taken by each of the processes of which the system is composed, especially the messages transmitted between them*
- *Communication Performance is a limiting characteristic*
  - *Latency, bandwidth, Jitter*
- *It is impossible to maintain a single notion of time*
  - *Computer clocks have drift*
  - *GPS: 1 micro Sec*
  - *Message Passing (eg.NTP) 100ms*

Communication speed

Compare:

```
static void Main(string[] args)
 {
        uint i = 0; long y; int x = 500;
        const uint MAX_ITER = 1000000; // 1million
        DateTime tmStart; DateTime tmEnd;

    tmStart = DateTime.Now;
    for (i = 0; i <= MAX_ITER; i++)
        y = x * i;
    tmEnd = DateTime.Now;
    TimeSpan tmDiff = tmEnd - tmStart;
    Console.WriteLine(MAX_ITER + "iterations took "+
                    (tmDiff.TotalMilliseconds ) + "ms");
  }
```

With:

Roundtrip time:

Ping www.cs

Ping krak.dk

Ping google.com

Ping www.jcu.edu.sg

Interaction model 1: Asynchronous systems

No known bounds for:

- The execution speed of a process
- Message delay on the network
- Clock drift

Interaction model 2: (Partly) Synchronous systems

- Known upper and lower bound for each process step
- Known upper bound for the time it task for a message to be received
- Known upper bound for clock drift

Failure Model

- The system might need to tolerate failures
  - processes
    - might stop / crash
    - degrade gracefully
    - exhibit Byzantine failures
  - may also be failures of
    - communication mechanisms

Omission and arbitrary failures

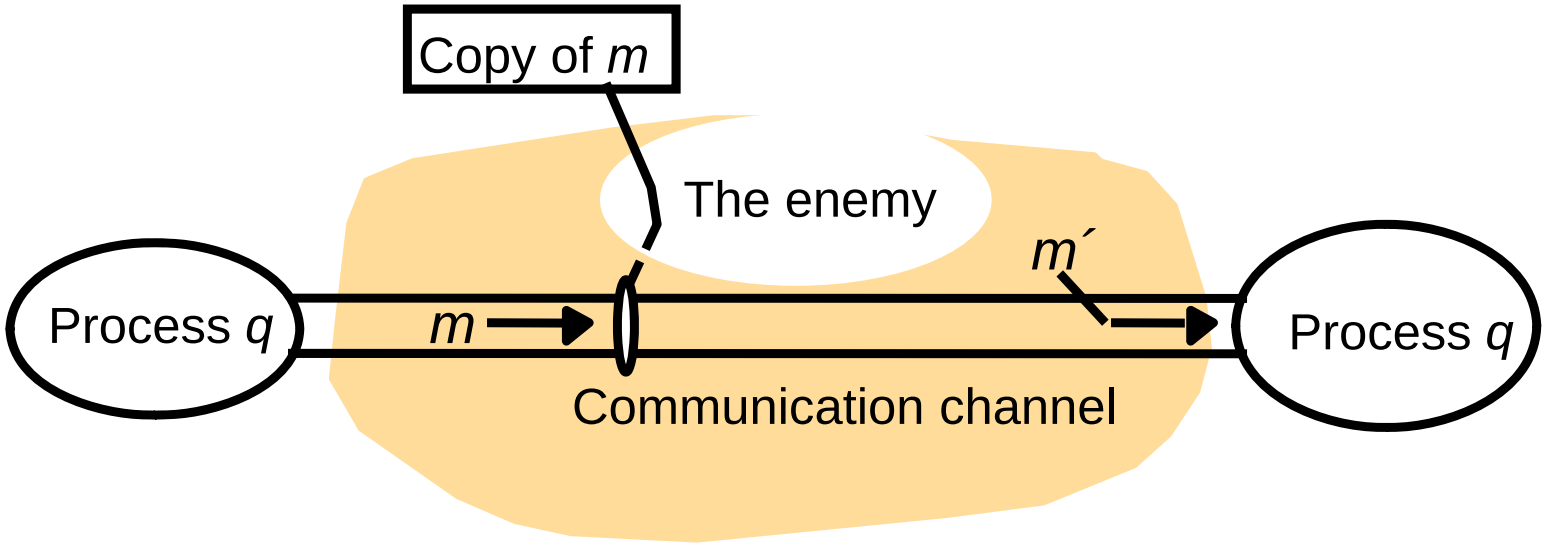
Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send* but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Timing failures

Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

Security model

- Protection of objects, ressources
- Securing processes and their interaction
  - Goals
    - Secrecy, integrity, authentication, authorization,..
  - Attacks
    - man-in-the-middle, eaves-dropping, play-back, …



END